운영체제 과제#3

20212908 이진

가산점 기능 구현 [✔]

test.c : 가산점 기능 테스트 프로그램

test.c, assignment3.c 파일 모두 sudo -s 명령어를 통해 실행

[가산점 커널 소스 파일 경로]

syscall_64.tbl : /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls

syscalls.h : /usr/src/linux/linux-5.15.120/include/linux

Makefile : /usr/src/linux/linux-5.15.120/kernel

fork.c : /usr/src/linux/linux-5.15.120/kernel

sys_print_cpu_burst.c : /usr/src/linux/linux-5.15.120/kernel

아래 파일들은 실제 과제 3 과는 관련 없는 시스템콜 함수이지만, 이전 과제 수행으로 인해 Makefile 에 등록했던 파일들이라 함께 첨부하였습니다.

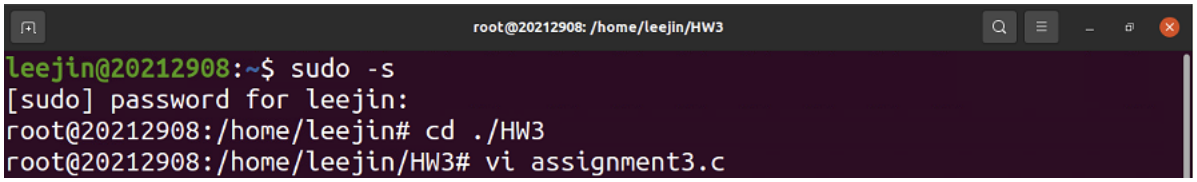
sys_print_hello.c : /usr/src/linux/linux-5.15.120/kernel

sys_reverse_order.c : /usr/src/linux/linux-5.15.120/kernel

sys_add.c : /usr/src/linux/linux-5.15.120/kernel

sys_sub.c : /usr/src/linux/linux-5.15.120/kernel

[시스템콜 코드 설명]



아래는 assignment3.c 에 대한 설명이다.

```c
#define _GNU_SOURCE         // GNU 확장 기능을 사용하도록 정의
#include <stdio.h>
#include <stdlib.h>         // exit
#include <unistd.h>         // fork, pipe, close
#include <time.h>           // clock_gettime
#include <sys/syscall.h>    // syscall
#include <sched.h>          // sched_setscheduler, sched_get_priority_max
#include <sys/resource.h>   // setpriority
#include <string.h>         // memset
#include <sys/wait.h>       // waitpid
#include <errno.h>          // perror
```

```c
// 각 자식 프로세스의 실행 결과를 출력
void printResult(pid_t pid, struct timespec start, struct timespec end, double elapsedTime, int nice) {
    // 프로세스 ID 출력
    printf("PID: %d ", pid);
    // CFS_NICE 일 때만 NICE 값 출력
    // CFS_NICE 가 아닐 때 nice == -1
    if (nice != -1) printf("| NICE : %d ", nice);

    char buf[100];
    struct tm tstart, tend;

    // 시작 시간을 로컬 시간 형식으로 변환
    localtime_r(&start.tv_sec, &tstart);
    // 종료 시간을 로컬 시간 형식으로 변환
    localtime_r(&end.tv_sec, &tend);

    // 프로세스 시작 시간 출력
    strftime(buf, sizeof(buf), "%H:%M:%S", &tstart);
    printf("| Start time: %s.%09ld ", buf, start.tv_nsec);

    // 프로세스 종료 시간 출력
    strftime(buf, sizeof(buf), "%H:%M:%S", &tend);
    printf("| End time: %s.%09ld ", buf, end.tv_nsec);

    // 소요 시간 출력
    printf("| Elapsed time: %f\n", elapsedTime);
    fflush(stdout);
}

// SCHED_PIPO, SCHED_RR 스케줄링 정책 설정
void setSchedulingPolicy(int policy, int priority, int timeSlice, int pid) {
    struct sched_param param;      // 스케줄링 매개변수 정의하기 위한 구조체
    param.sched_priority = priority;       // 프로세스 우선순위 설정

    // 스케줄링 정책 설정
    // pid : 변경하려는 프로세스의 id
    // policy : 적용하려는 스케줄링 정책 (SCHED_FIFO, SCHED_RR)
    if (sched_setscheduler(pid, policy, &param) == -1) {
        perror("sched_setscheduler");
        exit(EXIT_FAILURE);
```

```
        }
        // SCHED_RR 인 경우, time slice 설정
        if (policy == SCHED_RR) {
                FILE *fp;      //FILE 포인터 선언

                // 파일 이름 저장
                const char *filename = "/proc/sys/kernel/sched_rr_timeslice_ms";
                fp = fopen(filename, "w");      // 쓰기모드로 파일 열기
                if (fp == NULL) {
                        perror("Error opening file");      // fopen 실패 시 에러 출력
                        exit(EXIT_FAILURE);
                }

                fprintf(fp, "%d", timeSlice);             // 파일에 timeSlice 값을 정수로 기록
                fclose(fp);         // 파일 닫기
        }
}

int main() {
        // CPU 코어 개수 1 개로 제한
        // CPU 0 에 현재 프로세스를 바인딩
        cpu_set_t set;
        CPU_ZERO(&set);
        CPU_SET(0, &set);
        if (sched_setaffinity(0, sizeof(cpu_set_t), &set) == -1) {
                perror("sched_setaffinity");
                exit(EXIT_FAILURE);
        }

        // 스케줄링 정책 입력 받기 위한 문구 출력
        printf("Input the Scheduling Polity to apply:\n1. CFS_DEFAULT\n2. CFS_NICE\n3. RT_FIFO\n4. RT_RR\n0.
Exit\n");

        int pip[21][2];         // 파이프 배열 선언
        pid_t child_pid[21];       // 자식 프로세스 pid 값 저장하는 배열

        // 21 개의 Named_pipe(부모, 자식 프로세스 간 소통) 생성
        for (int i = 0; i < 21; i++) {
                if (pipe(pip[i]) == -1) {
                        perror("pipe");
```

```
            exit(EXIT_FAILURE);
        }
    }

    int option, nice = -1, timeSlice = -1, policy;
    // 스케줄링 정책 입력받기
    // 1. CFS_DEFAULT
    // 2. CFS_NICE
    // 3. RT_FIFO
    // 4. RT_RR
    scanf("%d", &option);
    if (!option) return 0;    // 0. exit
    if (option < 0 || option > 4) {        // 그 외 값 입력 시 종료
        return 0;
    }

    int priority = 0;

    // 입력 옵션에 따라 스케줄링 정책 및 우선순위 설정
    // CFS_DEFAULT 는 리눅스 스케줄링의 기본값이므로 따로 설정 X
    // CFS_NICE 는 기본 스케줄링에 nice 값만 변경하는 것이므로 따로 설정 X
    switch(option) {
        case 3:        //RT_FIFO
            policy = SCHED_FIFO;
            // sched_get_priority_max() : 특정 스케줄링 정책에 대해
            // 사용 가능한 최대 우선 순위 값 반환
            // 반환값 = 스케줄링 정책에서 사용할 수 있는 가장 높은 우선순위
            priority = sched_get_priority_max(SCHED_FIFO);
            break;
        case 4:        //RT_RR
            policy = SCHED_RR;
            priority = sched_get_priority_max(SCHED_RR);

            // RT_RR 의 경우, 적용할 timeSlice 도 입력 받기
            printf("Input the Time Slice to apply(10, 100, 1000): ");
            scanf("%d", &timeSlice);
            break;
    }

    // 작업 시작 시간, 종료 시간 저장하기 위한 구조체
```

```
struct timespec start, end;

for (int i = 0; i < 21; i++) {
        // 21 개의 자식 프로세스 생성
        child_pid[i] = fork();

        // CFS_NICE 의 경우 처음 7 개는 19, 다음 7 개는 0, 다음 7 개는 -20 으로 nice 값 변경
        if (option == 2) {
                if (i < 7) nice = 19;
                else if (i < 14) nice = 0;
                else nice = -20;
        }

        // 자식 프로세스 수행
        if (child_pid[i] == 0) {
                // CPU 코어 1 개로 제한
                // CPU 친화도 설정 sched_setaffinity(pid, 타입 크기 지정, 어느 cpu 코어에 할당할 것인지)
                if (sched_setaffinity(0, sizeof(cpu_set_t), &set) == -1) {
                        perror("sched_setaffinity");
                        exit(EXIT_FAILURE);
                }

                // RT_FIFO, RT_RR 정책의 경우 스케줄링 설정 함수 호출
                if (option == 3 || option == 4) {
                        setSchedulingPolicy(policy, priority, timeSlice, getpid());
                }
                // CFS_NICE 정책의 경우 기존 스케줄링 정책에서 nice 값 변경
                if (option == 2) {
                        // PRIO_PROCESS : 현재 프로세스의 우선순위를 변경하겠다는 것
                        // 0 : 현재 프로세스의 id
                        // nice : 지정할 nice 값, 낮을 수록 높은 우선 순위
                        if (setpriority(PRIO_PROCESS, 0, nice) < 0) {
                                perror("Fail setpriority");          // 실패할 경우 출력
                                exit(EXIT_FAILURE);
                        }
                }

                // 자식 프로세스 : 실행 시간 write 해야하니까
                // read 쪽 pip[i][0] 닫기
                if (close(pip[i][0]) == -1) {
```

```
            perror("Fail read pipe in child process");    //닫기 실패할 경우
            exit(EXIT_FAILURE);
        }

        // 현재 시간 측정 (시작 시간)
        clock_gettime(CLOCK_REALTIME, &start);

        // 배열 곱셈 연산 수행
        int count = 0, k, l, j;
        int result[102][102], A[102][102], B[102][102];
        memset(result, 0, sizeof(result));
        memset(A, 0, sizeof(A));
        memset(B, 0, sizeof(B));
        while(count < 100) {
            for (k = 0; k < 100; k++) {
                for(l = 0; l < 100; l++) {
                    for(j = 0; j < 100; j++) {
                        result[k][j] += A[k][l] * B[l][j];
                    }
                }
            }
            count++;
        }

        // 현재 시간 측정 (종료 시간)
        clock_gettime(CLOCK_REALTIME, &end);
        // 작업 수행하는 데 걸린 시간 초 단위로 계산
        double elapsedTime = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1000000000.0;

        // 수행 시간 파이프에 write
        // 부모 프로세스로 전송하기 위함
        write(pip[i][1], &elapsedTime, sizeof(elapsedTime));
        //write 이후 write 쪽 pip[i][1] 닫기
        close(pip[i][1]);

        // 결과 출력하는 함수 호출
        // 프로세스의 pid 값, 시작시간, 종료시간, 실행시간, nice 값 전달
        printResult(getpid(), start, end, elapsedTime, nice);

        // 성공적으로 작업 마친 후 자식 프로세스 종료
```

```
            exit(EXIT_SUCCESS);

        } else if (child_pid[i] < 0) {          // fork 실패할 경우 종료
            perror("fail fork");
            exit(EXIT_FAILURE);
        }
    }

    //부모 프로세스 수행
    double total_elapsed = 0;     // 모든 자식 프로세스 수행 시간의 합
    for (int i = 0; i < 21; i++) {
        int status;
        pid_t wpid;

        // 자식 프로세스 종료될 때까지 wait
        do {
            wpid = waitpid(child_pid[i], &status, 0);
        } while (wpid == -1);

        // 자식 프로세스가 정상적으로 종료되었는지 확인
        if (WIFEXITED(status)) {
            double elapsed;

            // 부모 프로세스 : 실행시간 read 해야하니까
            // write 쪽 pip[i][1] 닫기
            close(pip[i][1]);

            // 파이프를 통해 자식 프로세스의 실행 시간 읽기
            if (read(pip[i][0], &elapsed, sizeof(elapsed)) > 0) {
                total_elapsed += elapsed;
            } else {       // 실패 시 오류 메시지 출력 후 종료
                perror("Fail read in parent process");
                exit(EXIT_FAILURE);
            }
        }

        //read 이후 read 쪽 pip[i][0] 닫기
        close(pip[i][0]);
    }
```

```
    // 평균 실행 시간 = (자식 프로세스들의 실행 시간 총합) / (자식 프로세스 개수)
    double avgTime = total_elapsed / 21;

    // 적용된 스케줄링 정책 출력
    printf("Scheduling Policy: ");
    switch(option) {
        case 1:
            printf("CFS_DEFAULT | ");
            break;
        case 2:
            printf("CFS_NICE | ");
            break;
        case 3:
            printf("RT_FIFO | ");
            break;
        case 4:
            printf("RT_RR | ");
            break;
    }
    // RT_RR 의 경우 timeSlice 도 출력
    if (timeSlice != -1) printf("Time Quantum: %d ms | ", timeSlice);
    // 구해놓은 평균 실행 시간 출력
    printf("Average elapsed time: %f\n", avgTime);
}
```

[출력 결과]

1. CFS_DEFAULT

(제 컴퓨터 실행 환경에서는 3.6 초가 나와 cpu 코어 개수 제한이 잘 되지 않았나 싶어 다른 친구 환경에서 실행해보니 7 초 이상 나왔습니다!)

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
1
PID: 21658 | Start time: 13:46:04.296816047 | End time: 13:46:07.867748062 | Elapsed time: 3.570932
PID: 21651 | Start time: 13:46:04.267866950 | End time: 13:46:07.870040783 | Elapsed time: 3.602174
PID: 21648 | Start time: 13:46:04.252684771 | End time: 13:46:07.887528051 | Elapsed time: 3.634843
PID: 21643 | Start time: 13:46:04.236855961 | End time: 13:46:07.898630231 | Elapsed time: 3.661774
PID: 21652 | Start time: 13:46:04.272515539 | End time: 13:46:07.900081874 | Elapsed time: 3.627566
PID: 21639 | Start time: 13:46:04.264569863 | End time: 13:46:07.903746166 | Elapsed time: 3.639176
PID: 21647 | Start time: 13:46:04.250224522 | End time: 13:46:07.905739461 | Elapsed time: 3.655515
PID: 21638 | Start time: 13:46:04.300779527 | End time: 13:46:07.915517885 | Elapsed time: 3.614738
PID: 21649 | Start time: 13:46:04.256650635 | End time: 13:46:07.917233841 | Elapsed time: 3.660583
PID: 21656 | Start time: 13:46:04.288396542 | End time: 13:46:07.919474496 | Elapsed time: 3.631078
PID: 21644 | Start time: 13:46:04.240761460 | End time: 13:46:07.920124180 | Elapsed time: 3.679363
PID: 21646 | Start time: 13:46:04.248689276 | End time: 13:46:07.923242804 | Elapsed time: 3.674554
PID: 21650 | Start time: 13:46:04.260595700 | End time: 13:46:07.923611890 | Elapsed time: 3.663016
PID: 21645 | Start time: 13:46:04.244722247 | End time: 13:46:07.925816359 | Elapsed time: 3.681094
PID: 21641 | Start time: 13:46:04.228411716 | End time: 13:46:07.927123080 | Elapsed time: 3.698711
PID: 21653 | Start time: 13:46:04.276484449 | End time: 13:46:07.927458969 | Elapsed time: 3.650975
PID: 21642 | Start time: 13:46:04.232840485 | End time: 13:46:07.928416435 | Elapsed time: 3.695576
PID: 21657 | Start time: 13:46:04.292839174 | End time: 13:46:07.928826538 | Elapsed time: 3.635987
PID: 21640 | Start time: 13:46:04.226548779 | End time: 13:46:07.929164897 | Elapsed time: 3.702616
PID: 21654 | Start time: 13:46:04.276527263 | End time: 13:46:07.939052259 | Elapsed time: 3.662525
PID: 21655 | Start time: 13:46:04.284410462 | End time: 13:46:07.939695760 | Elapsed time: 3.655285
Scheduling Policy: CFS_DEFAULT | Average elapsed time: 3.652290
```

2. CFS_NICE

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
2
PID: 21601 | NICE : -20 | Start time: 13:43:53.321472507 | End time: 13:43:54.518346696 | Elapsed time: 1.196874
PID: 21603 | NICE : -20 | Start time: 13:43:53.336723171 | End time: 13:43:54.539253739 | Elapsed time: 1.202531
PID: 21602 | NICE : -20 | Start time: 13:43:53.328845439 | End time: 13:43:54.543480833 | Elapsed time: 1.214635
PID: 21604 | NICE : -20 | Start time: 13:43:53.345709556 | End time: 13:43:54.554305938 | Elapsed time: 1.208596
PID: 21605 | NICE : -20 | Start time: 13:43:53.356557745 | End time: 13:43:54.557338352 | Elapsed time: 1.200781
PID: 21606 | NICE : -20 | Start time: 13:43:53.372456381 | End time: 13:43:54.563982646 | Elapsed time: 1.191526
PID: 21607 | NICE : -20 | Start time: 13:43:53.380408950 | End time: 13:43:54.569434438 | Elapsed time: 1.189025
PID: 21597 | NICE : 0 | Start time: 13:43:53.309956780 | End time: 13:43:55.831703049 | Elapsed time: 2.521746
PID: 21600 | NICE : 0 | Start time: 13:43:53.320827222 | End time: 13:43:55.839758030 | Elapsed time: 2.518931
PID: 21594 | NICE : 0 | Start time: 13:43:53.302192926 | End time: 13:43:55.844687758 | Elapsed time: 2.542495
PID: 21596 | NICE : 0 | Start time: 13:43:53.308483025 | End time: 13:43:55.844963250 | Elapsed time: 2.536480
PID: 21598 | NICE : 0 | Start time: 13:43:53.312433316 | End time: 13:43:55.855781844 | Elapsed time: 2.543349
PID: 21595 | NICE : 0 | Start time: 13:43:53.304483804 | End time: 13:43:55.847167153 | Elapsed time: 2.542683
PID: 21599 | NICE : 0 | Start time: 13:43:53.315709879 | End time: 13:43:55.851223685 | Elapsed time: 2.535514
PID: 21588 | NICE : 19 | Start time: 13:43:53.292625204 | End time: 13:43:57.028247911 | Elapsed time: 3.735623
PID: 21592 | NICE : 19 | Start time: 13:43:53.300638689 | End time: 13:43:57.041969049 | Elapsed time: 3.741330
PID: 21593 | NICE : 19 | Start time: 13:43:53.302117402 | End time: 13:43:57.047199936 | Elapsed time: 3.745083
PID: 21591 | NICE : 19 | Start time: 13:43:53.296546965 | End time: 13:43:57.045545474 | Elapsed time: 3.748999
PID: 21590 | NICE : 19 | Start time: 13:43:53.300705526 | End time: 13:43:57.048257205 | Elapsed time: 3.747552
PID: 21589 | NICE : 19 | Start time: 13:43:53.301895438 | End time: 13:43:57.050124829 | Elapsed time: 3.748229
PID: 21587 | NICE : 19 | Start time: 13:43:53.289518805 | End time: 13:43:57.053795504 | Elapsed time: 3.764277
Scheduling Policy: CFS_NICE | Average elapsed time: 2.494108
```

CFS 는 각 프로세스에 가중치를 할당하여 동작한다. 이 가중치는 nice 값에 따라 결정되고, 이 값이 낮을 수록 더 높은 가중치를 받게 되어 우선순위가 높아진다. 이를 확인하기 위해 먼저 생성된 자식 프로세스에게 nice 값을 19, 그 후의 프로세스에게는 0, 그 후에는 -20 의 nice 값을 지정해줬다. 출력 결과를 보면 알 수 있듯이 nice 값이 작은 순서대로 프로세스가 실행되고 종료되고 있음을 알 수 있다. 또한, 모두 같은 작업을 하는 프로세스임에도 nice 값에 따라 Elapsed time 값이 다름을 알 수 있는데, 이는 nice 값이 작을 수록 해당 프로세스는 더 많은 CPU 시간을 할당받기 때문이다. CPU 시간을 많이 할당받으면 대기시간이 적어지고, 응답시간이 빨라지기 때문에 nice 값이 작을 수록 elapsed time 도 단축되는 결과를 얻을 수 있다.

+) CPU 코어 개수 제한 미적용 시 CFS_NICE (CPU_OnlyOne() 주석처리했을 때)

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
2
PID: 21553 | NICE : -20 | Start time: 13:43:00.844585933 | End time: 13:43:01.430161690 | Elapsed time: 0.585576
PID: 21551 | NICE : -20 | Start time: 13:43:00.829582939 | End time: 13:43:01.433334267 | Elapsed time: 0.603751
PID: 21557 | NICE : -20 | Start time: 13:43:00.868400975 | End time: 13:43:01.494524794 | Elapsed time: 0.626124
PID: 21556 | NICE : -20 | Start time: 13:43:00.864429645 | End time: 13:43:01.497214559 | Elapsed time: 0.632785
PID: 21554 | NICE : -20 | Start time: 13:43:00.856505078 | End time: 13:43:01.498917908 | Elapsed time: 0.642413
PID: 21552 | NICE : -20 | Start time: 13:43:00.833119705 | End time: 13:43:01.502289976 | Elapsed time: 0.669170
PID: 21555 | NICE : -20 | Start time: 13:43:00.862999078 | End time: 13:43:01.502405165 | Elapsed time: 0.639406
PID: 21548 | NICE : 0 | Start time: 13:43:00.829100994 | End time: 13:43:02.075243767 | Elapsed time: 1.246143
PID: 21547 | NICE : 0 | Start time: 13:43:01.280570423 | End time: 13:43:02.100440329 | Elapsed time: 0.819870
PID: 21550 | NICE : 0 | Start time: 13:43:00.829323848 | End time: 13:43:02.119495293 | Elapsed time: 1.290171
PID: 21545 | NICE : 0 | Start time: 13:43:00.840609256 | End time: 13:43:02.109720150 | Elapsed time: 1.269111
PID: 21544 | NICE : 0 | Start time: 13:43:00.822307209 | End time: 13:43:02.112089325 | Elapsed time: 1.289782
PID: 21549 | NICE : 0 | Start time: 13:43:00.824763616 | End time: 13:43:02.125534916 | Elapsed time: 1.300771
PID: 21546 | NICE : 0 | Start time: 13:43:01.276147643 | End time: 13:43:02.130478324 | Elapsed time: 0.854331
PID: 21540 | NICE : 19 | Start time: 13:43:00.822629670 | End time: 13:43:02.731235154 | Elapsed time: 1.908605
PID: 21543 | NICE : 19 | Start time: 13:43:00.836659689 | End time: 13:43:02.736131624 | Elapsed time: 1.899472
PID: 21539 | NICE : 19 | Start time: 13:43:00.825001788 | End time: 13:43:02.743011545 | Elapsed time: 1.918010
PID: 21537 | NICE : 19 | Start time: 13:43:00.822131574 | End time: 13:43:02.735354604 | Elapsed time: 1.913223
PID: 21541 | NICE : 19 | Start time: 13:43:00.828729018 | End time: 13:43:02.748364242 | Elapsed time: 1.919635
PID: 21538 | NICE : 19 | Start time: 13:43:00.832678911 | End time: 13:43:02.768748785 | Elapsed time: 1.936070
PID: 21542 | NICE : 19 | Start time: 13:43:00.833344799 | End time: 13:43:02.768919512 | Elapsed time: 1.935575
Scheduling Policy: CFS_NICE | Average elapsed time: 1.233333
```

3. RT_FIFO

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
3
PID: 21612 | Start time: 13:45:36.127130220 | End time: 13:45:36.307972656 | Elapsed time: 0.180842
PID: 21613 | Start time: 13:45:36.308188988 | End time: 13:45:36.482453343 | Elapsed time: 0.174264
PID: 21614 | Start time: 13:45:36.482707515 | End time: 13:45:36.657215464 | Elapsed time: 0.174508
PID: 21615 | Start time: 13:45:36.657647429 | End time: 13:45:36.834165286 | Elapsed time: 0.176518
PID: 21616 | Start time: 13:45:36.834406415 | End time: 13:45:37.010925993 | Elapsed time: 0.176520
PID: 21617 | Start time: 13:45:37.011156545 | End time: 13:45:37.190740845 | Elapsed time: 0.179584
PID: 21618 | Start time: 13:45:37.191014849 | End time: 13:45:37.367003738 | Elapsed time: 0.175989
PID: 21619 | Start time: 13:45:37.367260876 | End time: 13:45:37.544853976 | Elapsed time: 0.177593
PID: 21620 | Start time: 13:45:37.545164474 | End time: 13:45:37.720771578 | Elapsed time: 0.175607
PID: 21621 | Start time: 13:45:37.721623047 | End time: 13:45:37.948811590 | Elapsed time: 0.227189
PID: 21622 | Start time: 13:45:37.848484787 | End time: 13:45:38.120440242 | Elapsed time: 0.271955
PID: 21623 | Start time: 13:45:37.852430802 | End time: 13:45:38.297356876 | Elapsed time: 0.444926
PID: 21624 | Start time: 13:45:37.856416321 | End time: 13:45:38.466508722 | Elapsed time: 0.610092
PID: 21625 | Start time: 13:45:37.860849143 | End time: 13:45:38.642561947 | Elapsed time: 0.781713
PID: 21626 | Start time: 13:45:37.864818401 | End time: 13:45:38.918029917 | Elapsed time: 1.053212
PID: 21627 | Start time: 13:45:37.868788826 | End time: 13:45:39.092401423 | Elapsed time: 1.223613
PID: 21628 | Start time: 13:45:37.872764877 | End time: 13:45:39.261484319 | Elapsed time: 1.388719
PID: 21629 | Start time: 13:45:37.876724405 | End time: 13:45:39.431824179 | Elapsed time: 1.555100
PID: 21630 | Start time: 13:45:37.880696294 | End time: 13:45:39.612195039 | Elapsed time: 1.731499
PID: 21611 | Start time: 13:45:37.884662995 | End time: 13:45:39.791154500 | Elapsed time: 1.906492
PID: 21610 | Start time: 13:45:37.888641088 | End time: 13:45:40.024548619 | Elapsed time: 2.135908
Scheduling Policy: RT_FIFO | Average elapsed time: 0.710564
```

4. RT_RR

- 10ms

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
4
Input the Time Slice to apply(10, 100, 1000): 10
PID: 21662 | Start time: 13:46:27.119873535 | End time: 13:46:27.297132849 | Elapsed time: 0.177259
PID: 21663 | Start time: 13:46:27.297350383 | End time: 13:46:27.467071245 | Elapsed time: 0.169721
PID: 21664 | Start time: 13:46:27.467323322 | End time: 13:46:27.641483921 | Elapsed time: 0.174161
PID: 21665 | Start time: 13:46:27.641763972 | End time: 13:46:27.814665200 | Elapsed time: 0.172901
PID: 21666 | Start time: 13:46:27.814921384 | End time: 13:46:27.986913419 | Elapsed time: 0.171992
PID: 21667 | Start time: 13:46:27.987224656 | End time: 13:46:28.160031169 | Elapsed time: 0.172807
PID: 21668 | Start time: 13:46:28.160312429 | End time: 13:46:28.398923373 | Elapsed time: 0.238611
PID: 21669 | Start time: 13:46:28.192508603 | End time: 13:46:28.573154647 | Elapsed time: 0.380646
PID: 21670 | Start time: 13:46:28.196468131 | End time: 13:46:28.747669194 | Elapsed time: 0.551201
PID: 21671 | Start time: 13:46:28.200421572 | End time: 13:46:28.919378169 | Elapsed time: 0.718957
PID: 21672 | Start time: 13:46:28.204896896 | End time: 13:46:29.092576962 | Elapsed time: 0.887680
PID: 21661 | Start time: 13:46:28.208835677 | End time: 13:46:29.380651632 | Elapsed time: 1.171816
PID: 21673 | Start time: 13:46:28.212809664 | End time: 13:46:29.554170944 | Elapsed time: 1.341361
PID: 21674 | Start time: 13:46:28.216757360 | End time: 13:46:29.723098194 | Elapsed time: 1.506341
PID: 21675 | Start time: 13:46:28.220622901 | End time: 13:46:29.898342866 | Elapsed time: 1.677720
PID: 21676 | Start time: 13:46:28.224697466 | End time: 13:46:30.077061797 | Elapsed time: 1.852364
PID: 21677 | Start time: 13:46:28.228659513 | End time: 13:46:30.305778185 | Elapsed time: 2.077119
PID: 21678 | Start time: 13:46:28.232626095 | End time: 13:46:30.485186333 | Elapsed time: 2.252560
PID: 21679 | Start time: 13:46:28.236604413 | End time: 13:46:30.664772775 | Elapsed time: 2.428168
PID: 21680 | Start time: 13:46:28.240570758 | End time: 13:46:30.844399540 | Elapsed time: 2.603829
PID: 21681 | Start time: 13:46:28.244530445 | End time: 13:46:31.019929106 | Elapsed time: 2.775399
Scheduling Policy: RT_RR | Time Quantum: 10 ms | Average elapsed time: 1.119172
```

-100ms

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
4
Input the Time Slice to apply(10, 100, 1000): 100
PID: 21685 | Start time: 13:47:16.396150019 | End time: 13:47:16.573982156 | Elapsed time: 0.177832
PID: 21686 | Start time: 13:47:16.574188043 | End time: 13:47:16.743895955 | Elapsed time: 0.169708
PID: 21687 | Start time: 13:47:16.744131729 | End time: 13:47:16.914152190 | Elapsed time: 0.170020
PID: 21688 | Start time: 13:47:16.914468451 | End time: 13:47:17.086410219 | Elapsed time: 0.171942
PID: 21689 | Start time: 13:47:17.086622988 | End time: 13:47:17.270270951 | Elapsed time: 0.183648
PID: 21690 | Start time: 13:47:17.270550175 | End time: 13:47:17.451127880 | Elapsed time: 0.180578
PID: 21691 | Start time: 13:47:17.451806724 | End time: 13:47:17.627533411 | Elapsed time: 0.175727
PID: 21692 | Start time: 13:47:17.627761675 | End time: 13:47:17.806289619 | Elapsed time: 0.178528
PID: 21693 | Start time: 13:47:17.806542736 | End time: 13:47:17.983465175 | Elapsed time: 0.176922
PID: 21694 | Start time: 13:47:17.983726543 | End time: 13:47:18.292667261 | Elapsed time: 0.308941
PID: 21684 | Start time: 13:47:18.004849288 | End time: 13:47:19.548738171 | Elapsed time: 1.543889
PID: 21695 | Start time: 13:47:18.008777256 | End time: 13:47:19.611686422 | Elapsed time: 1.602909
PID: 21696 | Start time: 13:47:18.012729997 | End time: 13:47:19.673804446 | Elapsed time: 1.661074
PID: 21697 | Start time: 13:47:18.016693416 | End time: 13:47:19.737756069 | Elapsed time: 1.721063
PID: 21698 | Start time: 13:47:18.020716724 | End time: 13:47:19.801285223 | Elapsed time: 1.780568
PID: 21699 | Start time: 13:47:18.024652091 | End time: 13:47:19.863892778 | Elapsed time: 1.839241
PID: 21700 | Start time: 13:47:18.028647352 | End time: 13:47:19.928693393 | Elapsed time: 1.900046
PID: 21701 | Start time: 13:47:18.032616118 | End time: 13:47:19.999474210 | Elapsed time: 1.966858
PID: 21702 | Start time: 13:47:18.036596699 | End time: 13:47:20.122960014 | Elapsed time: 2.086363
PID: 21703 | Start time: 13:47:18.040532876 | End time: 13:47:20.191306192 | Elapsed time: 2.150773
PID: 21704 | Start time: 13:47:18.044496375 | End time: 13:47:20.256565886 | Elapsed time: 2.212070
Scheduling Policy: RT_RR | Time Quantum: 100 ms | Average elapsed time: 1.064700
```

-1000ms

```
leejin@20212908:~/HW3$ sudo ./assignment3
Input the Scheduling Polity to apply:
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. Exit
4
Input the Time Slice to apply(10, 100, 1000): 1000
PID: 21734 | Start time: 13:48:14.390979625 | End time: 13:48:14.566274958 | Elapsed time: 0.175295
PID: 21735 | Start time: 13:48:14.566500457 | End time: 13:48:14.738906566 | Elapsed time: 0.172406
PID: 21736 | Start time: 13:48:14.739603435 | End time: 13:48:14.910148750 | Elapsed time: 0.170545
PID: 21737 | Start time: 13:48:14.910468343 | End time: 13:48:15.081583337 | Elapsed time: 0.171115
PID: 21738 | Start time: 13:48:15.081846486 | End time: 13:48:15.252826005 | Elapsed time: 0.170980
PID: 21739 | Start time: 13:48:15.253067599 | End time: 13:48:15.424965539 | Elapsed time: 0.171898
PID: 21740 | Start time: 13:48:15.425182279 | End time: 13:48:15.651303229 | Elapsed time: 0.226121
PID: 21741 | Start time: 13:48:15.588595980 | End time: 13:48:15.826266280 | Elapsed time: 0.237670
PID: 21742 | Start time: 13:48:15.592529222 | End time: 13:48:16.004820661 | Elapsed time: 0.412291
PID: 21733 | Start time: 13:48:15.596496921 | End time: 13:48:16.193121236 | Elapsed time: 0.596624
PID: 21743 | Start time: 13:48:15.600465545 | End time: 13:48:16.371142646 | Elapsed time: 0.770677
PID: 21744 | Start time: 13:48:15.604463170 | End time: 13:48:16.644960613 | Elapsed time: 1.040497
PID: 21745 | Start time: 13:48:15.608400315 | End time: 13:48:16.819025293 | Elapsed time: 1.210625
PID: 21746 | Start time: 13:48:15.612864080 | End time: 13:48:16.997322934 | Elapsed time: 1.384459
PID: 21747 | Start time: 13:48:15.616563694 | End time: 13:48:17.180524284 | Elapsed time: 1.563961
PID: 21748 | Start time: 13:48:15.620807412 | End time: 13:48:17.360138983 | Elapsed time: 1.739332
PID: 21749 | Start time: 13:48:15.624796686 | End time: 13:48:17.539771035 | Elapsed time: 1.914974
PID: 21750 | Start time: 13:48:15.628762708 | End time: 13:48:17.784391025 | Elapsed time: 2.155628
PID: 21751 | Start time: 13:48:15.632700554 | End time: 13:48:17.966982550 | Elapsed time: 2.334282
PID: 21752 | Start time: 13:48:16.541085295 | End time: 13:48:18.148928555 | Elapsed time: 1.607843
PID: 21753 | Start time: 13:48:16.544856986 | End time: 13:48:18.332336850 | Elapsed time: 1.787480
Scheduling Policy: RT_RR | Time Quantum: 1000 ms | Average elapsed time: 0.953081
```

[가산점]

1. RT_FIFO 스케줄링 정책 변경하기

```
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi fork.c
```

리눅스의 기본 스케줄링 정책은 CFS 이다. 실시간 스케줄링으로는 SCHED_FIFO, SCHED_RR 을제공한다. kernel/fork.c 파일은 fork 시스템 호출을 구현하는 코드를 포함하고 있다. fork 명령어를 통해 자식 프로세스를 생성했을 때, 스케줄링 정책을 SHED_FIFO 로 변경하기 위해 해당 파일을 수정한다.

root@20212908: /usr/src/linux/linux-5.15.120/kernel

```
        retval = copy_fs(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_files;
        retval = copy_sighand(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_fs;
        retval = copy_signal(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_sighand;
        retval = copy_mm(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_signal;
        retval = copy_namespaces(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_mm;
        retval = copy_io(clone_flags, p);
        if (retval)
                goto bad_fork_cleanup_namespaces;
        retval = copy_thread(clone_flags, args->stack, args->stack_size, p, args
->tls);
        if (retval)
                goto bad_fork_cleanup_io;

        p -> policy = SCHED_FIFO;

        stackleak_task_init(p);

        if (pid != &init_struct_pid) {
                pid = alloc_pid(p->nsproxy->pid_ns_for_children, args->set_tid,
                                args->set_tid_size);
                if (IS_ERR(pid)) {
                        retval = PTR_ERR(pid);
                        goto bad_fork_cleanup_thread;
                }
        }

        /*
         * This has to happen after we've potentially unshared the file
         * descriptor table (so that the pidfd doesn't leak into the child
         * if the fd table isn't shared).
         */
        if (clone_flags & CLONE_PIDFD) {
                retval = get_unused_fd_flags(O_RDWR | O_CLOEXEC);
                if (retval < 0)
                        goto bad_fork_free_pid;

                pidfd = retval;

"fork.c" 3218L, 80095C                                2211,26-33    69%
```

p 는 task_struct 구조체의 포인터로 선언된 변수이다. task_struct 는 리눅스 커널에서 프로세스의 상태를 나타내는 구조체이다. 프로세스의 pid, 실행 상태, 스케줄링 정보, 파일디스크립터, 신호처리 등의 정보를 포함하고 있다. 자식 프로세스가 부모 프로세스로부터 fork 됐을 때, 리눅스 커널은 시스템 호출을 통해 task_struct 를 복사하여 새로운 프로세스를 생성한다. 이 때, 자식 프로세스는 부모 프로세스의 스택 메모리를 그대로 사용하지 않기 때문에 스택 메모리를 초기화 해야한다. 해당 함수가 바로 stackleak_task_init(p)이다. 그렇기에 스택 메모리를 초기화하는 작업 전에 생성된 프로세스의 스케줄링 정책을 변경하는 코드를 삽입하였다. task_struct 구조체의 policy 필드는 해당 프로세스의 스케줄링 정책을 나타낸다. 해당 필드를 SCHED_FIFO 값으로 할당해준다.

2. CPU burst 누적값 커널 로그에 출력하기

CPU burst 누적값은 task_struct 구조체에서 얻을 수 있다. 유저영역에서 소비된 CPU 시간을 나타내는 utime 과, 커널 영역에서 소비된 CPU 시간을 나타내는 stime 을 더하면 된다. 해당 값은 task_struct 의 필드로 저장되어 있다. 이를 통해 시스템콜 함수를 등록하여 유저영역에서 생성된 자식 프로세스의 pid 값을 인자로 보내고 시스템콜 함수에서 task_struct 구조체의 값을 얻어내서 커널 로그에 출력할 것이다.

```
root@20212908:/home/leejin/HW3# cd /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls
root@20212908:/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls# vi syscall_64.tbl
```

먼저, 시스템콜 테이블을 등록한다.

```
447     common  memfd_secret            sys_memfd_secret
448     common  process_mrelease        sys_process_mrelease
449     common  print_hello             sys_print_hello
450     common  reverse_order           sys_reverse_order
451     common  add                     sys_add
452     common  sub                     sys_sub
453     common  print_cpu_burst         sys_print_cpu_burst

#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64.  These syscalls have numbers 512-547.
# Do not add new syscalls to this range.  Numbers 548 and above are available
# for non-x32 use.
#
-- INSERT --                                                    377,48-60     90%
```

453 번에 cpu burst 누적값을 커널 로그에 출력하는 시스템콜 함수인 sys_print_cpu_burst 를 등록하였다.

```
root@20212908:/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls# cd ../../../../include/l
inux
root@20212908:/usr/src/linux/linux-5.15.120/include/linux# vi syscalls.h
```

해당 함수를 시스템콜 헤더 파일에 등록한다.

```
asmlinkage long long sys_print_cpu_burst(const int);

#endif
-- INSERT --                                                    1390,53       Bot
```

asmlinkage 를 앞에 붙힘으로써 어셈블리 코드에서도 C 함수 호출이 가능해진다.
인자로는 프로세스의 pid 값을 받고, 성공여부는 0(성공), -1(실패) 값으로 리턴해주는 함수의 프로토타입을 정의한다.

```
root@20212908:/usr/src/linux/linux-5.15.120/include/linux# cd ../../kernel
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_print_cpu_burst.c
```

시스템콜 함수를 구현한다. 파일명은 상관없지만, 파일 내부의 함수를 sys_시스템콜 이름으로 작성해야 한다. 아래는 sys_print_cpu_burst.c 코드의 설명이다.

root@20212908: /usr/src/linux/linux-5.15.120/kernel

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/sched.h>
#include <linux/sched/signal.h>
#include <linux/pid.h>

asmlinkage long long sys_print_cpu_burst(const int pid) {
        struct pid *pid_struct;
        struct task_struct *task;
        pid_struct = find_get_pid(pid);
        if (!pid_struct)
                return -1;
        task = pid_task(pid_struct, PIDTYPE_PID);
        if (!task) {
                printk(KERN_INFO "PID %d Fail", pid);
                put_pid(pid_struct);
                return -1;
        }

        unsigned long long total_utime = task -> utime;
        unsigned long long total_stime = task -> stime;
        unsigned long long total_time_in_sec = (total_utime +  total_stime) / HZ;
        printk(KERN_INFO "[pid : %d] CPU burst : %llu\n", pid, total_time_in_sec);

        put_pid(pid_struct);
        return 0;
}
SYSCALL_DEFINE1(print_cpu_burst, int, pid) {
        return sys_print_cpu_burst(pid);
}
~
~
~
-- INSERT --                                                    21,32-39      All
```

```
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi Makefile
```

추가한 시스템콜이 다른 시스템콜과 함께 컴파일될 수 있도록 Makefile 을 수정한다.

```
obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o capability.o ptrace.o user.o \
            signal.o sys.o umh.o workqueue.o pid.o task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o ucount.o regset.o sys_print_hello.o \
            sys_reverse_order.o sys_add.o sys_sub.o sys_print_cpu_burst.o
```

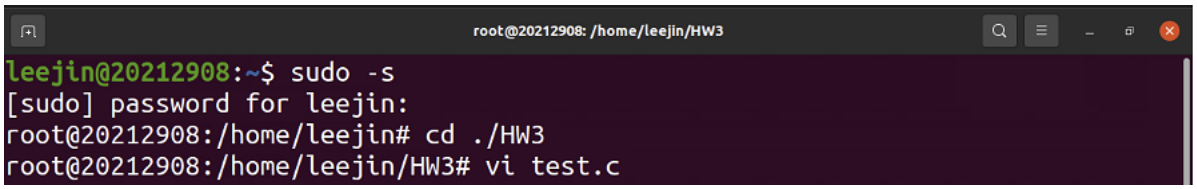
sys_print_cpu_burst.o 를 추가해주었다.

```
root@20212908:/usr/src/linux/linux-5.15.120/kernel# cd ..
root@20212908:/usr/src/linux/linux-5.15.120# make-kpkg --J 2 --initrd --revision=4.0 kernel_image
root@20212908:/usr/src/linux/linux-5.15.120# cd ..
root@20212908:/usr/src/linux# ls
linux-5.15.120         linux-image-5.15.120_2.0_amd64.deb  linux-image-5.15.120_4.0_amd64.deb
linux-5.15.120.tar.xz  linux-image-5.15.120_3.0_amd64.deb
root@20212908:/usr/src/linux# dpkg -i linux-image-5.15.120_4.0_amd64.deb
```

.커널 소스 디렉토리로 이동하여 새로 컴파일한 후 재부팅을 실시한다. 기존의 revision 값과 겹치지 않게 4.0 이라는 다른 정수값을 할당하여 컴파일을 실시해주었다.

```
done
root@20212908:/usr/src/linux# reboot
```

3. 테스트 프로그램 작성



아래는 fork()를 통해 자식 프로세스를 21 개 생성하고 배열 곱셈 연산을 수행한다. 연산이완료되면, 이전에 만들어둔 sys_print_cpu_burst 시스템콜을 호출하여 커널 로그에 cpu burst 값을 출력한 후 종료하는 프로그램이다.

```
root@20212908:/home/leejin/HW3# gcc test.c -o test
```

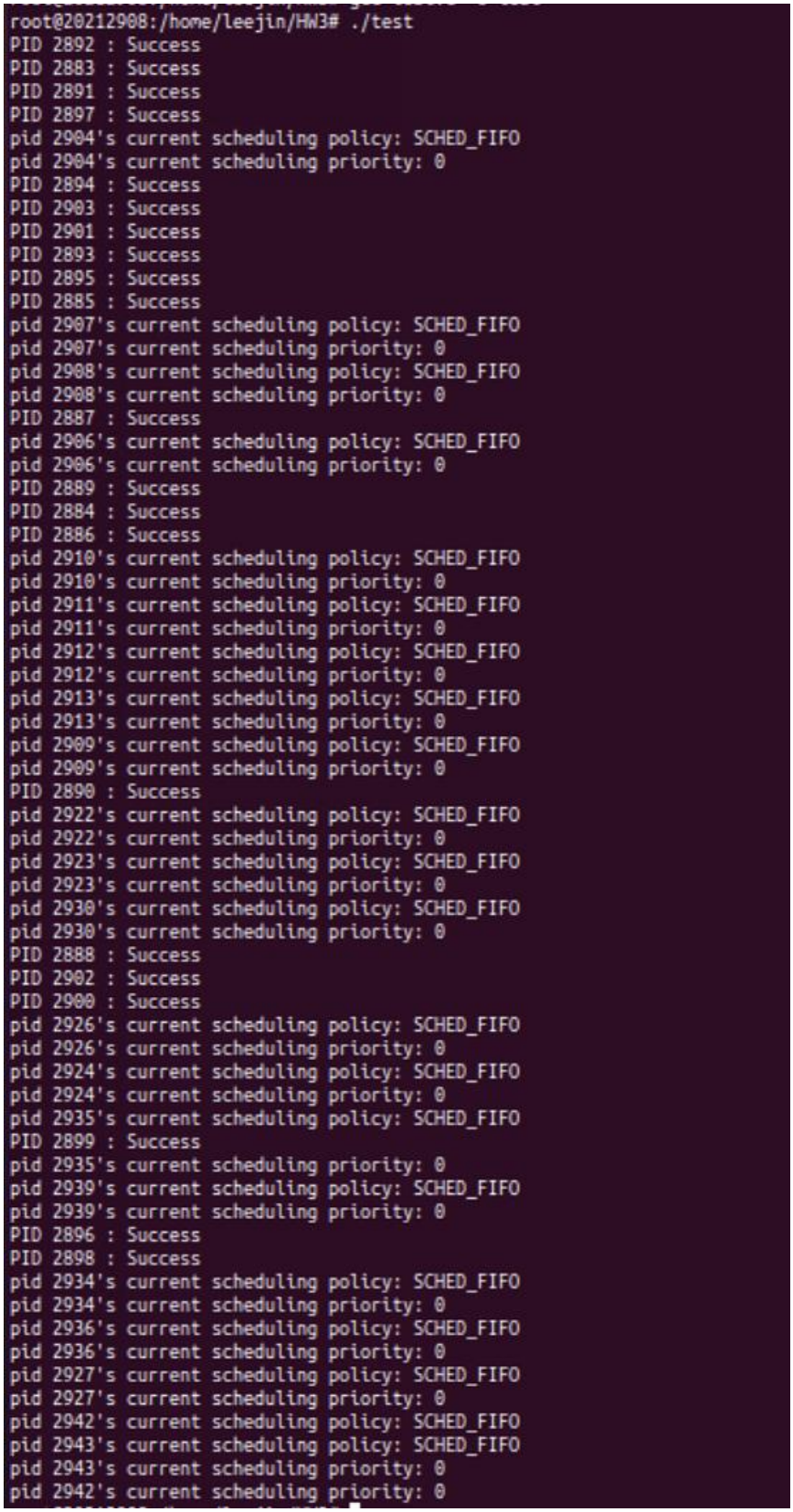
컴파일 해준다. 아래는 test.c 코드에 대한 설명이다.

```c
#include <stdio.h>
#include <stdlib.h>
// 표준 라이브러리 헤더 파일 (exit)
#include <unistd.h>
// POSIX 운영 체제 API 헤더 파일 (fork)
#include <sys/wait.h>       // 자식 process 끝날 때까지 기다리는 wait 함수
#include <string.h>         // 문자열 처리 함수를 위한 헤더 파일 (memset)
#include <linux/kernel.h> // 리눅스 커널을 위한 헤더 파일
#include <sys/syscall.h>   // 시스템 콜 호출을 위한 헤더 파일

int main() {
    pid_t child_pid[21]; // 자식 프로세스의 PID 를 저장할 배열 선언

    for (int i = 0; i < 21; i++) { // 21 개의 자식 프로세스를 생성
        child_pid[i] = fork(); // 자식 프로세스 생성
        if (child_pid[i] < 0) { // fork 가 실패한 경우
            perror("Fail fork");
            exit(EXIT_FAILURE);
        }

        if (child_pid[i] == 0) { // 자식 프로세스

            // 배열 곱셈 수행
            int count = 0, k, l, j;
            int result[102][102], A[102][102], B[102][102];
            memset(result, 0, sizeof(result));
            memset(A, 0, sizeof(A));
            memset(B, 0, sizeof(B));
            while(count < 100) {
                for (k = 0; k < 100; k++) {
                    for(l = 0; l < 100; l++) {
                        for(j = 0; j < 100; j++) {
                            result[k][j] += A[k][l] * B[l][j];
```

```
                        }
                    }
                }
                count++; // 반복 횟수 증가
            }

            if (syscall(453, (int)getpid()) == 0) { // 시스템 콜 453 을 호출
                printf("pid %d : Success", (int)getpid()); // 성공 메시지 출력
                system("chrt -p $$");   // 어떤 스케줄링 정책 사용하는지 출력
            }
            exit(EXIT_SUCCESS); // 자식 프로세스 종료
        }
    }

    // 부모 프로세스
    for (int i = 0; i < 21; i++) {
        waitpid(child_pid[i], NULL, 0); // 각 자식 프로세스의 종료할 때까지 대기
    }

    return 0;
}
```

4. 출력 결과

프로그램이다 시스템콜 호출을 성공하면 해당 프로세스의 pid 값과 함께 Success 가 출력된다. 또한, system("chrt -p $$")를 사용하여 해당 프로세스가 어떤 스케줄링 정책을 사용하는지도 명시적으로 출력해주었다.



```
root@20212908:/home/leejin/HW3# dmesg
```

커널 로그에 잘 출력되었는지 확인하기 위해 dmesg 명령어를 사용한다.

```
[ 1435.375448] [pid : 2897] CPU burst : 768000
[ 1435.385205] [pid : 2892] CPU burst : 752000
[ 1435.386602] [pid : 2883] CPU burst : 768000
[ 1435.389888] [pid : 2901] CPU burst : 768000
[ 1435.397151] [pid : 2893] CPU burst : 768000
[ 1435.399155] [pid : 2895] CPU burst : 784000
[ 1435.404688] [pid : 2891] CPU burst : 768000
[ 1435.405678] [pid : 2894] CPU burst : 784000
[ 1435.407022] [pid : 2903] CPU burst : 768000
[ 1435.410517] [pid : 2885] CPU burst : 784000
[ 1435.412605] [pid : 2887] CPU burst : 752000
[ 1435.417653] [pid : 2889] CPU burst : 752000
[ 1435.421141] [pid : 2886] CPU burst : 752000
[ 1435.427530] [pid : 2902] CPU burst : 752000
[ 1435.430612] [pid : 2884] CPU burst : 784000
[ 1435.432181] [pid : 2890] CPU burst : 768000
[ 1435.435329] [pid : 2888] CPU burst : 752000
[ 1435.435761] [pid : 2900] CPU burst : 768000
[ 1435.439369] [pid : 2899] CPU burst : 784000
[ 1435.440230] [pid : 2896] CPU burst : 752000
[ 1435.441972] [pid : 2898] CPU burst : 768000
[38177.124118] [pid : 3606] CPU burst : 736000
[38177.125090] [pid : 3601] CPU burst : 752000
[38177.130695] [pid : 3619] CPU burst : 752000
[38177.136194] [pid : 3618] CPU burst : 736000
[38177.141863] [pid : 3620] CPU burst : 736000
[38177.142537] [pid : 3617] CPU burst : 752000
[38177.142736] [pid : 3616] CPU burst : 752000
[38177.145496] [pid : 3614] CPU burst : 752000
[38177.150805] [pid : 3605] CPU burst : 752000
[38177.155997] [pid : 3608] CPU burst : 720000
[38177.172549] [pid : 3602] CPU burst : 768000
[38177.173743] [pid : 3615] CPU burst : 736000
[38177.176962] [pid : 3612] CPU burst : 736000
[38177.177882] [pid : 3607] CPU burst : 736000
[38177.182238] [pid : 3613] CPU burst : 752000
[38177.183184] [pid : 3621] CPU burst : 736000
[38177.186220] [pid : 3609] CPU burst : 752000
[38177.186945] [pid : 3604] CPU burst : 768000
[38177.190975] [pid : 3611] CPU burst : 768000
[38177.193180] [pid : 3610] CPU burst : 768000
[38177.193591] [pid : 3603] CPU burst : 752000
root@20212908:/home/leejin/HW3#
```

21 개의 자식 프로세스 pid 의 cpu burst 누적값이 정상적으로 커널 로그에 출력됐음을 확인할 수 있다.